DOCUMENT RESUME

ED 282 544 IR 012 693

AUTHOR Donkin, John Caldwell
TITLE Optimization Methods for Computer Animation.
PUB DATE 86
NOTE 77p.; Master's Thesis, The Ohio State University.
PUB TYPE Dissertations/Theses - Master Theses (042)

EDRS PRICE MF01/PC04 Plus Postage.
DESCRIPTORS *Animation; *Computer Graphics; Cost Effectiveness; *Efficiency; Motion; Problem Solving; Programing; Three Dimensional Aids
IDENTIFIERS Optimization; Story Boards

ABSTRACT

Emphasizing the importance of economy and efficiency in the production of computer animation, this master's thesis outlines methodologies that can be used to develop animated sequences with the highest quality images for the least expenditure. It is assumed that if computer animators are to be able to fully exploit the available resources, they should have a working understanding of computer programming methods. The animation problems that are presented and explored are solved from an animator's viewpoint, but solutions to some of the problems used as examples were arrived at through computer programming techniques. The thesis is divided into six chapters: (1) Computer Graphics Overview; (2) Storyboards: The Animation Guideline; (3) Analyses and Customization; (4) Motion Control Systems; (5) Reflections, Highlights, Glows, Glints, and Shadows; (6) Natural Phenomena; and (7) Conclusions. Eleven figures are included in the text, examples of thumbnail sketches and timing sheets are provided in three appendices, and 32 references are listed. (MES)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED282544

OPTIMIZATION METHODS FOR COMPUTER ANIMATION

A Thesis

Presented in Partial Fulfillment of the Requirements for the degree of Master of the Arts in the Graduate School of the Ohio State University

by

John Caldwell Donkin, B.F.A.

* * * * *

The Ohio State University

1986

Master's Examination Committee:

Thomas E. Linehan

Charles A. Csuri

Approved by

Thomas E. Linehan

Advisor
Department of Art Education

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

John C. Donkin

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

IR013693

BEST COPY AVAILABLE

THESIS ABSTRACT

THE OHIO STATE UNIVERSITY
GRADUATE SCHOOL

NAME: Donkin, John Caldwell QUARTER/YEAR: Fall 1986

DEPARTMENT: Art Education DEGREE: Masters of Arts

ADVISOR'S NAME: Linehan, Thomas E.

TITLE OF THESIS: Optimization Methods for Computer Animation

This thesis emphasizes the importance of economy and efficiency in the production of computer animation. It also outlines methodologies which can be used to develop animated sequences. The thesis assumes the position that computer animators, in order to be able to fully exploit the available resources, should have a working understanding of computer programming methods. The problems that are presented and explored in this thesis are solved from an animators viewpoint, but are solved from the above position. The goal of the methodologies presented is to put forth the highest quality images as possible, but at the lowest possible resource expenditure.

Thomas E. Linehan
Advisor's Signature

Copyright (c) 1986
by John Caldwell Donkin
All rights reserved.

ACKNOWLEDGEMENTS

I would like to acknowledge the support of my advisor, Tom Linehan and Charles Csuri, the director of The Computer Graphics Research Group (CGRG). Their support and encouragement have been strongly felt in my experience at CGRG. I would like to thank my family for fostering my creative ambitions. I would also like to thank my creative friends who have inspired me to strive for new heights in computer animation: to John Berton whose personal work is an inspiration and who first introduced me to computer animation; to Jeff Light for his support and willingness to discuss any topic under the sun; to Susan Van Baerle who always has a unique perspective to offer; to Anne Seidman who is a great friend and is consistent in supporting me and my work; and, to Teresa Cavanaugh for her unending support and friendship. I would also like to acknowledge the influence of Frank Thomas and Ollie Johnston, whose dedication to the art of animation is an incredible inspiration. Their enthusiasm for new technology is an encouraging embrace. I can only hope that I will be able someday to bring as much life to computer animation as they did to a bunch of filmed drawings.

VITA

October 26, 1961.....	Born - Corning, New York
June 1983.....	B.F.A., Photography and Cinema, The Ohio State University, Columbus, Ohio
1984 - Present.....	Technical Director/Animator, Cranston/Csuri Productions, Columbus, Ohio
Field of Study:	Computer Graphics and Animation.

TABLE OF CONTENTS

Acknowledgements.....ii

Vita.....iii

Table of Contents.....iv

List of Figures.....v

Introduction.....1

Chapter 1: Computer Graphics Overview.....5

Chapter 2: Storyboards: The Animation Guideline.....11

Chapter 3: Analyses and Customization.....16

Chapter 4: Motion Control Systems.....28

Chapter 5: Reflections, Highlights, Glows, Glints and Shadows.....34

Chapter 6: Natural Phenomena.....50

Chapter 7: Conclusion.....60

Appendices.....62
Appendix A: Thumbnail Sketches.....62
Appendix B: Timing Sheet.....63
Appendix C: Timing Sheet.....64

List of References.....65

LIST OF FIGURES

Figure 1. The coordinate system.....7

Figure 2. Scene from "Trash".....24

Figure 3. Linear distance versus time graph (DVT)...31

Figure 4. Ease-in and Ease-out DVT graph.....31

Figure 5. Variable acceleration DVT graph.....31

Figure 6. Linear and splined interpolation.....33

Figure 7. Reflective world.....37

Figure 8. Bounding box configuration.....38

Figure 9. Highlight object.....46

Figure 10. Scene from "Trash".....49

Figure 11. Probability curve and random distribution.....59

INTRODUCTION

Computer animation has come a long way since its beginnings as a scientific innovation. Computer animation is now a vital part of television graphics, and commercial advertisements. It is also becoming an important part of the feature film industry. More impressively it is beginning to impact on the more general realm of animation. It is now common to find computer generated entries to animation festivals around the world. Computer animation is breaking out of the mold of being interesting only to other computer animators. One of the major reasons for this insurgence into society is that computer systems are becoming more accessible to artists for use for artistic means. As computer technology continues to reduce in price it becomes more available to schools as well as individuals.

Computer animation is still used scientifically however. New techniques for rendering realistic imagery are still in the forefront of the industry's innovations. This points up two basic approaches to producing computer graphics animation: the scientific

and the intuitive. The scientific approach is the attempt to approach realistic images incorporating a methodology which is based on scientific laws relevant to the desired effect. The intuitive approach is to gain the desired effect through any means, without strict regard to known scientific laws.

One major difference between these two approaches is the way in which they are interpreted. The scientific approach is by and large designed to develop imagery which is a simulation of an event or portions of an event which occur in reality. The intuitive approach does not attempt to simulate reality. It does however attempt to achieve imagery which is of the desired quality. It in fact may be quite realistic. Many things can be simulated quite well by scientific means. This fact is quite evident in images which are rendered using a technique called "ray-tracing". A drawback to using methods like ray-tracing is that these images tend to be expensive to calculate due to the complexity of the mathematical formulas from which they were derived. As a result it is often the case that it is difficult, if not impossible, to use such techniques for non-scientific applications. This fact is slowly

changing though with the advent of faster and less expensive computing devices.

Intuitive methods are generally less expensive to calculate but they require that the animator put a large amount of effort into developing techniques which are specific to the problem at hand. While not absolutely non-scientific, intuitive methods are designed to evoke an emotional response to the image. The image is supposed to "look" and "feel" right as opposed to "be" right. Intuitive methods are designed to maximize image quality within the given constraints of the system that is being used to develop the animation. These constraints may include limited resource availability, limits on image complexity and computational expense. This study is designed to give animators and software developers insight into how they can employ these methods in order to maximize economy without sacrificing image quality. This will involve the development of solutions to various animation problems through the use of examples. The solutions presented are intended to provide the readers with practical solutions to real problems, but more importantly to provide them with a methodology to incorporate with any animation problem.

In addition, this study will discuss the importance of isolating and avoiding the problems early. This is done by careful use and development of the animation storyboard.

There has long been a controversy over whether a computer animator should have the ability to develop software programs, or if they should be just a "user". It is my opinion that while an animator is not required to have extensive programming experience, some programming knowledge is highly recommended. The main reason I assume take this position is that an animator is constantly challenged with new problems which have not always been worked out previously by a programmer. It helps for the animators to have programming knowledge so that they can either solve the problem themselves, or provide a programmer with information which can be used to solve the problem quickly. As a result, some solutions which I provide in this thesis are achieved through programmatical methods.

CHAPTER 1

Computer Graphics Overview

Before I go any further, it makes sense to clear up some terminology. The following is a brief description of what computer generated imagery is, and some of its basic concepts. Perhaps the first thing to point out is that the emphasis of this thesis will be based on three-dimensional computer graphics.

Three-dimensional animation is defined as sequential graphical images where the original database is composed of three-dimensional "objects" which are placed within a three-dimensional space. This space is called the "world space". These objects are then transformed to a two-dimensional space, called the "screen space". This is analogous to a camera, which takes a two-dimensional picture of an actual three dimensional scene. As with a real camera, computer animators can move the imagined camera within the world space. The world space is defined in somewhat mathematical terms. A point in space can be uniquely

defined by locating its position along three pre-defined axes. The locations on the axes are referred to as the "coordinates" of that point. These axes, by convention, are called the X-axis, the Y-axis, and the Z-axis. The X Y and Z axes are joined together at an "origin" whose coordinates are defined to be (0, 0, 0). The three axes are perpendicular to each other. In this way we define a coordinate system which can be used to determine the exact location of objects, or points in space. If the "camera" is located on the positive Z-axis, looking towards the origin, the axes are defined as follows: the positive X-axis points to the right, the negative X-axis points to the left. The positive Y-axis points up, while the negative points down. The positive Z-axis points towards the camera, while the negative points away from the camera (figure 1). It is the motion of objects within this coordinate system that is the basis for computer animation.

In order to move objects about on the screen it becomes important to understand what objects are, and how they are made. There are several ways in which objects can be made, and represented. Perhaps the most common method, and the one assumed in this paper, is to

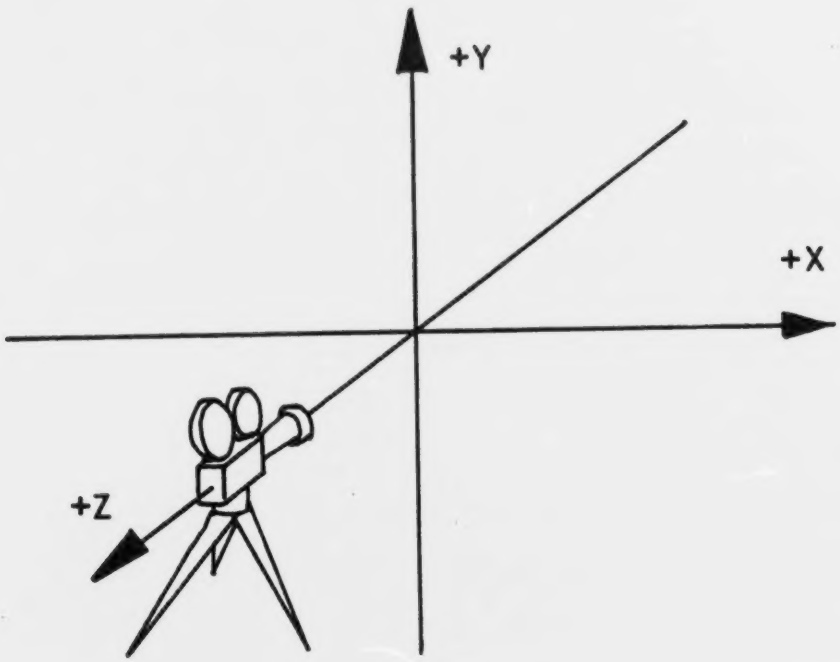


Figure 1.

The Coordinate System.

describe objects which are composed of points and polygons. A point is a location in space as defined by its position in the coordinate system in world space (as described above). A polygon is simply a surface which is described by a plane which passes through three or more points. The points which describe this plane also define the edges of the polygon, which in turn determines the

shape of the polygon. The points which define the edges of a polygon are also called the "vertices" and are attributed to that polygon. Vertices from one polygon may be shared by one or more polygons. In the case of a cube, there are 6 polygons, and 8 points. Each polygon has 4 vertices, 4 edges and is square in shape.

In addition to objects which are built and manipulated there are other types of objects which are used in the animation of a scene. One of these objects has already been mentioned: the camera. In describing a camera there are a number of parameters which must be specified in order to accurately simulate a real camera. Perhaps the most obvious of these parameters is the specification of the location of the camera in space. This, like other objects, is specified as an XYZ coordinate. The next most obvious parameter is to have some way of describing the orientation of the camera; the camera must be directed towards something. There are basically two methods to do this. While the methods are different, it is possible to inter-mix them in a single animation environment, in that the results from one method can be derived from the other.

The first method requires that in addition to

positioning the "eyepoint" (another name for the location of the camera), the "center of interest" must also be positioned. The center of interest is the point at which the camera is always directly pointed. Moving the center of interest results in the camera changing its direction.

The other method is to control the camera in the same way that a real camera is controlled. This is done by specifying pan and tilt information. A pan is a back and forth motion (similar to turning your head). The tilt is an up and down motion (nodding). These motions are specified in degrees from a standard direction. Both of these methods require specifying the roll angle. The roll angle is the angle of rotation of the camera about the axis that is defined by the direction of the camera. This is easily envisioned by tilting the head from side to side.

Another parameter that is necessary to complete the camera model is the "view angle" of the camera. The view angle is analogous to the focal length of the lens. By changing the view angle it is possible to create the effect of a zoom.

Other special objects include the light sources.

Some systems allow for very complex methods of specifying light sources. These parameters can include: position, range, direction, spread (diffusion), and color.

While this has been a quick introduction to computer animation it should be enough to begin discussion into the first phase of developing a computer animation sequence. An excellent overview of three dimensional computer animation can be found in Judy Sachter's "The Basic Concepts of Three-Dimensional Computer Graphics For Artists". [22]

CHAPTER 2

Storyboards: The animation guideline

A storyboard is basically a description of the action that is to take place on the screen. The storyboard, its traditions stemming from those of film and cel animation, has played a large part in the generation of computer generated animation. The storyboards, simply put, are a drawn version of the animation action accompanied by a written description of the scene. The description may include the dialogue and an indication of the sound effects which may be an important part of the scene.

There are some very important factors to consider when developing a storyboard, as well as when one is being analyzed and interpreted. Storyboards should be a clear representation and guide to the action that is going to take place. There are an infinite number of ways that a storyboard can be made depending upon its particular application. One common bond that they all

share is that they should be made in a way that is appropriate for their particular function.

There are basically two types of storyboards: the presentation storyboard and the animation storyboard. The distinction between these two types of "boards" can be summarized as that one board is designed to used by a person or persons who will not be doing animation (the presentation board), while the other is designed to be used by a person who will be doing animation (the animation storyboard).

The immediate question that is inferred is: "Why would somebody want to look at a storyboard who will not be doing animation?" This can be answered rather quickly when the assumption is made that computer animation is extremely expensive, and often animators are not substantially wealthy enough to afford to fund their own projects. Because of this fact it can be very important to have a storyboard that can clearly portray the animator's artistic ideas to some funding person or organization.

In the case of commercial or commissioned work, it is important that a client be able to clearly understand the storyboard so that they know that the

animation is going to meet their specific needs. For this reason the presentation board should be made to be seen and understood in and of itself. It should not be dependent upon an "explanation" by an animator. It may be the case (and often is) that the board has to be made available to someone who is not going to be available to the animator for discussion. It is a difficult thing to determine how detailed to make a storyboard for presentation. The storyboard should not be so close to the final animation that it takes longer to create the storyboard than it does to develop the animation. But the should be clear enough to give the viewer a good idea of what to expect from the final product.

The animation storyboard should be used as a tool by the animator. It is an inexpensive way to experiment with new and different ideas. The animation storyboard need not be a storyboard in the same sense as the presentation board, although the presentation board can be used in conjunction with the animation board. The animation board's major function is to serve as an aid in developing the animation, and ways to produce it. As such, the animation storyboard can consist of several different representational approaches, whatever makes

sense for a particular scene. This can come in the form of "thumbnails" , which are described by Frank Thomas and Ollie Johnston in their book The Illusion of Life [24] as "little exploratory sketches". They advise using these "before doing anything else, to make sure that everything will stage well and will look as convincing as the animator had hoped." While they are referring to hand drawn animation, it is the design of the animation that they are referring to and this applies to computer animation as easily as it does to drawn animation. Examples of thumbnail sketches can be found in appendix A.

It may be useful to sketch out other types of motions as well. It may be useful to sketch motion paths for objects as an aid to doing the animation. The animation storyboard may be accompanied by single frame renderings for color and data checking, model sheets and story sketches. These sketches can be combined with timing sheets as an aid to producing the animation (see appendices B and C).

It is not uncommon for a storyboard to be filmed or videotaped to be composited into a "story reel". The story reel is edited to fit to the actual timing of the

animation. It may include dialogue and sound effects which will actually appear in the final animation. This serves as an excellent aid to the overall timing and pacing of the animated sequence, and may provide information which may not be found by any other means.

CHAPTER 3

Analyses and Customization

I mentioned earlier that intuitive methods require that animators have the ability to derive solutions to the problems at hand. In effect, these solutions are a customization which Turner Whitted claims "is the key to economy" [29]. Customization involves modifying the parameters of an algorithm to match the special cases that are introduced by the application. This has the unfortunate drawback that a solution to a problem in one situation may not be generalized to include all cases of that problem. It does however make it easier to find a solution since all the special cases do not need to be solved.

The first stage in producing animation is to carefully examine the storyboard for the sequence that is to be animated and to determine the main motivation for that sequence. Once this is done a sequence breakdown should be done. The sequence breakdown is a

way of breaking the scene down into a series of isolated elements. These can then be prioritized in terms of their importance in the entire scene as an aid to the animator.

An animator should work from the general to the specific. The first priority in the scene should be those elements which create the general timing and feel of the scene. The details should specifically be left out at this stage. This will allow the animator to quickly identify problems which may be a result of original design flaws of the scene. This will also allow the animator freedom to change the scene without the expense of changing all of the detail which may be dependent upon the more general motion.

In this way the animation is built up in a "building block" fashion [24], where each element is built upon the previous element. The overall effect of this technique is that the animation will tend to take its general form quite quickly. The detail will come more slowly as each element is eventually added. As the final elements are added, the changes in the scene will be increasingly subtle, perhaps nearly imperceptible, yet each adds to the total effect, and each compliments

each other.

-- Data Generation

A strong advantage to working in this way is that it is less likely that "over-detailing" will occur. Over-detailing occurs when the rendered resolution is not capable of reproducing the complexity of the data. This results in loss of the detail in the rendered image. The most frequent occurrence of over-detailing is when a complex model is seen in the screen from a viewpoint which is far away. As with real objects the farther the distance the object is from the viewer the smaller the apparent size of the image. This smaller apparent size results in the object being rendered in a relatively small number of pixels. This points up the importance of having objects built with the complexity and detail that is appropriate for the scene. If it is known that an object is to be viewed from long distance, then a less complex model is needed. If the object is to be seen in a close-up then a more complex model is needed.

The time that it takes to generate the object data is not the only factor to be considered when determining data complexity. Rendering time tends to go up as data becomes complex. It makes sense to have simplified versions of data for those scenes where the lack of complexity will not be seen. It may be the case that a particular object will be seen in several different resolutions. The object data in this case should be constructed in varying complexity or multiple levels of detail. It may be possible to switch from one level of detail to another within the course of a single scene. This may be done in one of two ways: an instantaneous transition, or a blended transition [17]. The instantaneous substitution is a simple swap as a function of the number of pixels that the object covers [17, 2]. The Blended transition is an interpolative technique for the transition between levels of detail.

Even if an object is to be seen in only one general resolution it can be beneficial to have simplified versions of the data available for low-resolution motion testing. The low-resolution motion tests are a way for animators to preview animation without the expense of rendering full images. Since the high level of detail

would be lost in the motion tests simplified data should be used, unless the test is designed to test some particular aspect of the data set itself.

As for the generation of object there are several methods which have been substantially defined [31], these methods include: projection, surface of revolution, tubular object generators [25], and lofting techniques. For objects which change shape over time there is a less defined method for altering the shapes. There are basically two techniques that are used in the motion of dynamically changing objects.

The two techniques are quite similar in their approach, the only real difference is in the way in which they are applied. The first method is that of shape interpolation. Shape interpolation is the technique that has been used quite effectively in such animated films such as "Snoot and Muttly" [26], "Uneven Bars" [27], "Trash" [8] and in "Tony DePeltrie" [1]. The basic idea behind shape interpolation is that there are certain "key" objects which determine the way in which the objects changes shape. Point number one in object A corresponds to point number one in object B. The interpolation occurs when an intermediate object is

produced whose points are somewhere in between the location of the points in either of the "key" objects. [25]. A constraint in using interpolation is that the "key" objects must have the same data configuration. It is for this reason that the "key objects" should be constructed in a way that is parametric and repeatable. The programs which generate the objects should take input parameters which control the type of object that is generated. The program should be repeatable so that the configuration of the individual "key objects" is consistent. The least reliable method for assuring object consistency is to build the data "by hand".

The other technique for doing dynamically changing data is by strict replacement of the object on a per/frame basis. The way that this is most typically done is that the objects are generated every frame by some object generating program. This method does not have the constraint that the objects be configured in the exact same way, however consistency is important to provide a smooth transition over a series of frames. A factor to consider when using this method for animation is the time in which it takes to compute the data. This time must be included in the rendering time for the

animation as it is required every frame. It may be possible if system space allows it to pre-compute the object descriptions and simply index through them as the images are computed. If sufficient space is available this is the method that should be used because the data will only have to be generated once. This may save time if several motion tests are made.

-- Rendering Considerations

Once the data has been created it is important to determine the optimum method for rendering each element in a scene. Richard Chueng in "Rendering for Television" [5] cites some guidelines which are useful:

-Composite elements whenever possible
- Render at the lowest resolution possible
- Render only the elements that are moving
- Render with the lowest resolution texture possible

These guidelines are all followed in an effort economize

rendering.

Compositing of images can be used to reduce rendering time substantially. Often in the case of commercial work there is a need to produce several versions of a single animation. These versions may have variations which can be accomodated simply if a layering technique is employed. An common example is one where a particular animation may need to be presented over two or more different backgrounds. It is then the case that the foreground objects can be separated from the background and composited over separate backgrounds. This means that the separate backgrounds may only need to be calculated for each version. If the foreground objects were calculated with the background in the frame this would not be possible.

This ties in with the guideline which states "only calculate objects that are moving". If there is a static element in a sequence, this should be singled out as a separate layer and calculated once for the entire sequence. This element can then be composited with other moving elements to yield the final scene. An excellent example of how this works can be found in the film "Trash". There is a scene in which there is a view up at

a building (figure 2). The clouds are moving in the background, there is a neon light which is flashing on a sign attached to the building. A light comes on in an apartment window. All of these elements were calculated as separate layers and combined together to yield the final composite.

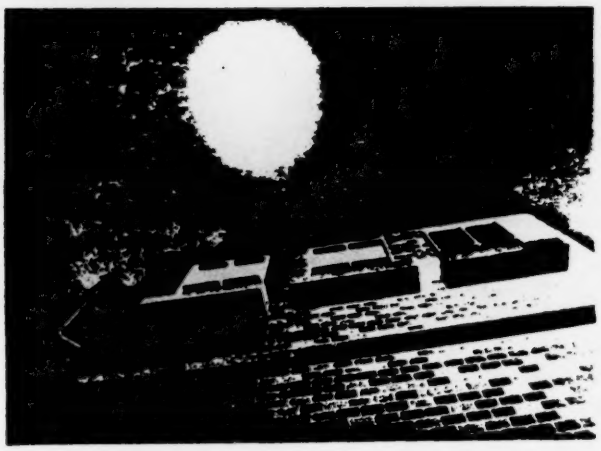


Figure 2.

Scene From "Trash".

The scene was built from back to front. The background elements consisted of a crescent moon and a glow around the moon. These two elements were combined together to be the first layer. This layer was then combined with a moving layer of the clouds moving. These clouds were calculated in low resolution and blown up to the final resolution. Then there is the non-moving layer of the foreground building. The building in the foreground was fairly expensive to calculate as it was texture mapped and covered a large area of the screen. This layer was calculated once for the entire scene. The next layer was the flashing neon sign. The neon sign was calculated once for each of its different letters in the word which appears on the sign. The windows were calculated twice; once with the lights on and once with them off. Finally the window frames were only calculated once. The results of this scene were quite good and it was only possible as a result of saving the calculation time in this way.

The clouds in the previous sequence were rendered at a quarter of the normal resolution and were blown-up to the full resolution. This was done both as a

calculation savings and for aesthetic reasons. Since clouds are a soft entity, the gentle blurring effect that happens as a result of increasing the resolution was not deemed a problem, and in fact is desirable. For soft or blurry objects, things such as glows or objects which are going to be blurred, should be calculated at lower resolutions. Since the detail in the image is designed to be lost, there is no sense in calculating the detail in the first place.

Similarly lower resolution texture maps can be used to save calculation of objects as well. Low resolution texture maps can be used to nice effect under certain circumstances. Mip mapping is a technique which allows for automatic usage of lower resolution texture maps according to the needs of the scene. Mip maps also result in improved image quality as a result of loss of detail from long distances [30, 6]. The building in the previously mentioned example was texture mapped using a mip mapping technique.

If possible special purpose rendering algorithms should be used for objects which meet the criteria that they require. Special algorithms can be developed to do a variety of things. For instance neon is a light source

and as such would have no surface shading. The neon in the previously mentioned scene was rendered using a display program where no shading is applied (the image was still antialiased). This turns out to be a useful algorithm for computing shadows, mattes and light sources. It runs quite a bit faster than the standard display program. Other display algorithms can be developed to render object primitives [2]. An algorithm to quickly render shaded geometric objects such as spheres [12] can be useful in certain situations. These special programs can be used not only to improve the speed at which things calculate but can actually be used to enhance the image quality.

CHAPTER 4

Motion Control Systems

The control of motion of objects is the new frontier facing animators and programmers in computer animation. As the display drivers reach an equalibrium there will be an increasing need for software to produce animation with increasingly complex motion. There are two general types of motion in animation. There is the motion of monolithic objects [23] in space, and there is motion through modeling, which is to say that the motion is the result of the data actually changing. I previously had described two types of motion through modeling, that of shape interpolation and data swapping and have more discussions about dynamic data in chapter 6 (Natural Phenomena).

The motion of objects in three-dimensional computer animation still has a long way to go. Several animations systems have been developed which are adequate for many types of motion of objects. These include "Twixt version

3" developed by Julian Gomez while at The Ohio State University [10], ASAS [21], PODA [16] and Wavefront [28] which is a fairly new release of software which is commercially available.

A method which seems to work quite well is to develop a motion path through space of a particular object. This path is a path of an attribute of the object only, it contains no information about the value of that attribute at any given time. To determine the value of the attribute at a given point in time a second curve is used. In the case of the positional attribute, this curve represents distance over time. That is to say that it is a graph of the distance that the object has travelled over the course of a given amount of time (figure 3). If the object is to move in a linear fashion along the path the distance versus time (DVT) [15] graph will be a straight line from a distance 0 to the total distance. A curved DVT path will yield a speed differential along the path. As the DVT path flattens horizontally the speed is lessened, as the curve slopes upward the speed increases (figures 4 and 5). By using a graph of this type to control the speed and position of objects along paths it is possible

to get smooth transitions between "Key" locations. It is also possible to guarantee that an object will be at a specific location along the path at a very specific time.

It is not only the location of objects that can be controlled in this fashion. The values for virtually any track [10] of an object can be controlled in this way. It could be the rotation of the object or some other attribute of the objects such as color, illumination parameters or scale.

The strength of incorporating a working model of this type is that the animation is not bound to time. The tracks are prepared independently from the time. If the timing for a scene needs to be changed but the path is to remain the same, the DVT curve would be used to represent a different period of time, thus changing the overall timing of the move.

This technique also allows for intuitive control of the acceleration and deceleration of a track. A graph of the acceleration and deceleration can simply be hand drawn, and played back, altered and played back again. It also allows for continuity or discontinuity (if its desired) of the rate at which an attribute changes.

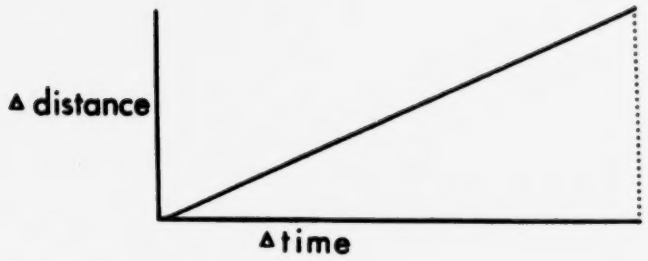


Figure 3.

Linear distance versus time graph (DVT).

Figure 4.

Ease-in and ease-out DVT graph.

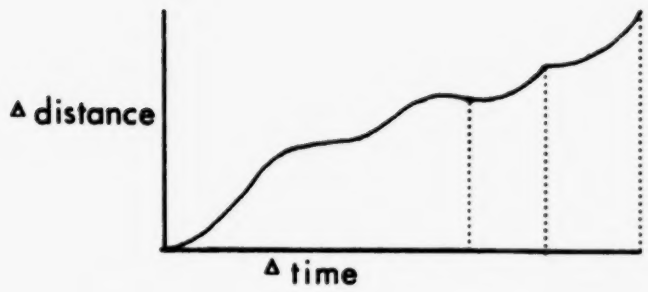


Figure 5.

Variable acceleration DVT graph.

Current methods for institution of this technique is as a special program which generates new path files which can be applied to the various attributes. It would however be quite compatible with some of the existing animation systems and could be easily implemented fully as a feature.

-- Shape Interpolation

Currently most methods of shape interpolation is linear. The result of linear shape interpolation is somewhat limited. Either a large number of "key shapes" are required to avoid some of the anomalies that it can incur [25] or the motion just looks linear. Another type of interpolation between shapes is splined interpolation [14]. In linear interpolation one object transforms from one object to another in a linear fashion. If the location of a point in object A is connected to the same point in object B with a straight line, this is the path that the resultant point will take. In splined interpolation the paths that the points would take are

splined together to form a smoothly curved path from one location to the other. This path is computed for every point in the object. Each path is determined by the corresponding points for each of the "key objects" (figure 6). Once the path is computed the point that is the appropriate distance along that path is output to form a new object. This distance is determined as a fractional portion of the total distance along the path. Each point that is output to the new data set is located as a result of the same fractional proportion of the path along which it is travelling. This technique yields rather nice results, however it does tend to be quite expensive to compute the new data set.

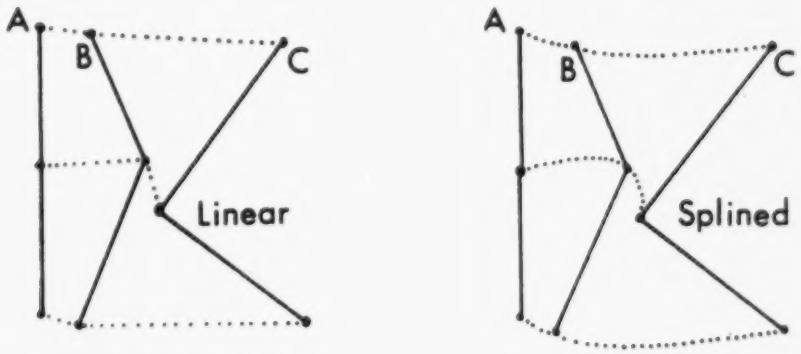


Figure 6.

Linear and splined interpolation.

CHAPTER 5

Reflections, Highlights, Glows, Glints and Shadows

The importance of attention to detail is one of the single most important concepts to grasp when dealing with animation. Often it is the small details that make a piece of animation stand out. People who work on special effects in theatrical films have a saying. Paraphrased, it goes something like this: "If you see it, then it isn't good enough". What is meant by this saying is that the special effects have to be so well done that the audience doesn't realize that the effect is a trick. It should happen naturally in the scene and should not be noticed. Nothing sticks out in a scene more than a special effect done poorly. If the effect is not there however, the audience will notice that something is missing.

Rarely is it the case in computer graphics (at this stage in its infancy) that an effect has to match so seamlessly with live action that the audience isn't

supposed to notice. It is important though for computer generated effects to seem natural. It is in the area of special effects that "tricks" play an important role in achieving the desired effect. Some of the more common effects are: reflections, glows, light washes (or glints), highlights, and lighting effects.

Given the problem of reflections we can apply the methodology presented thus far. The first step is to analyze what reflections are and how the should look on the screen. In order to do this it is helpful to understand some principles of reflections. Objects in reality which reflect other objects or their environment can be made of many different materials. The reflective quality of different materials are not the same. Computer generated objects can also have different surface qualities which can give an audience an indication of the type of surface that the object is supposed to represent. Similarly if the reflective quality of an object does not match the type of object that it is intended then the wrong impression may be left with the audience. Therefore the decisions that are made regarding the reflecting surface must be made with consideration to the application of the storyboard.

In analyzing a reflection from a flat mirror, the overriding analyses shows that the reflection is for all practical purposes perfect. The image that is reflected in the mirror is of the same quality as the environment that is outside the mirror. Objects which have a less clear image reflected are taken to be less mirror-like, or less reflective. What can be concluded from this is that highly reflective surfaces have sharper reflections than dull or flat surfaces.

As an example of how to actually generate a reflection for a scene, let us assume that a scene calls for a bottle to be placed upon a shiny surface such as a glass coffee table.

The first step is to break the shot down into all of its elements. The distinguishing element in this scene is that the scene calls for a reflection. The data for the scene is fairly simple and should not require any special attention.

Suppose the initial data is already built, that is to say that the bottle and the table in which it is reflected are built. The next step is to set up the scene using these two elements. The scene is now set. The table is in frame and the bottle is sitting on the

table. The next stage is to begin work on the reflection.

A reflection on a flat surface such as a table like this is the result of some very well known scientific principles. But the results of those principles are as follows: the reflected image is the same as if there were a real scene on the other side of the reflecting plane and the viewer is looking through a window into that other world (figure 7). The difference being that the objects are all reversed from what they are on this side of the reflecting plane. The technique that will apply in this scenario is to produce a scene which is that which would be seen in that "reflective world".

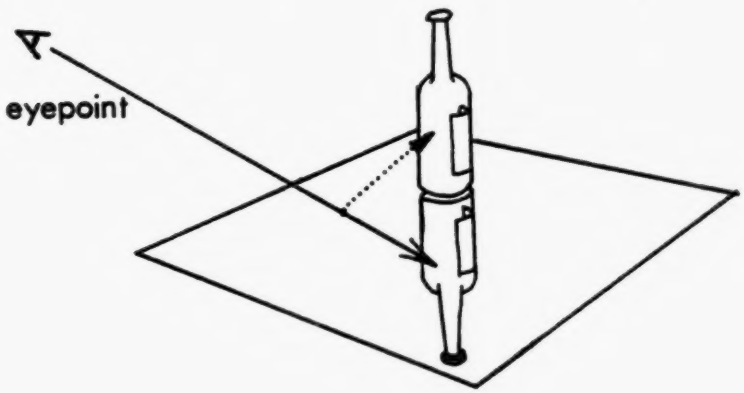


Figure 7.

Reflective world.

Fortunately this reflective world is based on the non-reflective world and there is a simple transformation that can be used to derive this world from the original scene. The reflected object (the object in the reflection) is derived from the reflecting object (the object that is being reflected on the surface) by scaling the reflecting object in the Y direction by -1.0 (for horizontal reflections). This has the result of making all of the positive Y values of the object negative and vice versa. The reflected object should then be transformed to have the upper limit of its bounding box in the Y axis equal to the minimum limit of the bounding box in the Y axis of the reflecting object (figure 8).

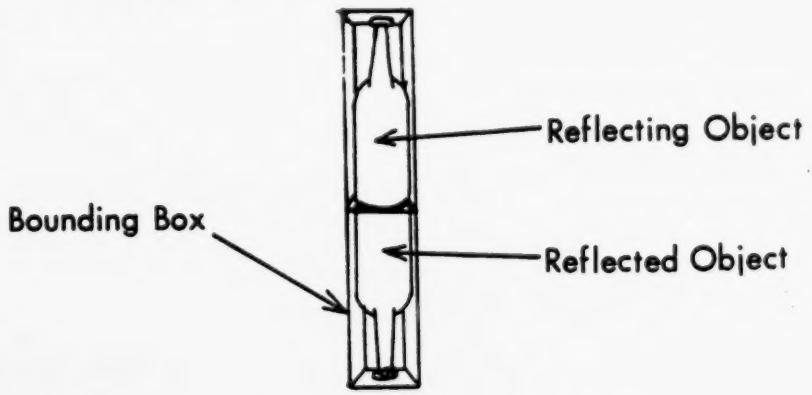


Figure 8.

Bounding box configuration.

This is done to allow for an easy transformation of the objects orientation in the scene. The transformations of the reflected object on a horizontal plane are derived from the orientation of the reflecting object using the following equations:

Position:

```
PXr = Pxo
PYr = PYp - (PYo - PYp)
PZr = PZo
```

Rotation:

```
RXr = RXo * -1.0
RYr = RYo
RZr = RZo * -1.0
```

Scale:

```
SXr = SXo
SYr = SYo
SZr = SZo
```

Where the 'r' subscript refers to the reflected object the 'o' subscript refers to the original object (reflecting) and the 'p' subscript refers to the Y position of the reflecting plane.

If these transformations are performed the reflected object will appear in the correct location relative to the reflecting object and the reflecting

surface. The equations are derived from an intuitive point of view. The X and Z coordinates of the location of the object are the same for both the reflecting and reflected objects. The position in Y are the inverse of each other relative to the reflecting plane. The scaling in the X and Z are equivalent as is the Y component. (remember a negative scale was already applied to derive the reflected object in the first place.) The rotational transformations are intuitive as well. Rotations in the X and Z axes are reversed while the Y rotation is the same.

-- Calculation

The three elements that need to be calculated are:

A.) The reflected image (the reflected bottle).
B.) The reflecting objects (the bottle on the table).
C.) The reflecting surface (the table).

These elements should be calculated separately and composited into the final image later. This is done so that the images can be combined in a way that is convincing. In calculating the reflected image it is

important to remember that all light sources should be considered as objects, and as such should have the reflecting transformations applied to them when the reflected image is calculated. Also it is important that if there is more than one object being reflected that these objects be calculated together or should be composited together into a single layer before final compositing. The reasons for this will be evident later in this chapter.

The earlier discussions about the importance of separating layers can be seen well in this example. The reflecting object should be calculated as a separate element. This is because the reflected object should only be seen in the areas covered on the screen by the reflecting surface. This adds to the believability of the reflection. It also provides the animator with a mechanism of for altering the reflective quality of the reflecting surface, in that the image produced as the reflected image can be altered using post-processing such as blurring.

-- Compositing the image

The order in which the elements are composited are important in developing the proper illusion. The first stage is to display the reflecting plane. The reflected image should then be displayed on the screen, but only in the areas covered by the reflecting plane in screen space. It is at this stage that the reflection can offer information about the type of material that composes the reflecting surface. If the reflected image is shown nearly opaque and with crisp edges then the reflecting plane will appear to be quite shiny. If the reflection is shown with much transparency or blurring is introduced, the surface will appear to be more dull. Finally the reflecting object should be displayed.

When the reflected objects are displayed over the reflecting surface it is important that all of the objects to be reflected are calculated together or are previously composited together. If the reflection is to be shown with transparency the entire reflecting image should be shown transparently, not each element in the

reflected image. If each layer of the reflected element are shown individually with transparency, objects which are supposed to be behind other objects will show through. This is not the effect that is expected in a realistic reflection. It is important to remember that the reflected image is a "window on another world" and should be correct.

Some details which can be introduced when considering reflections are:

- The color of the surface, and how that may affect the color of the reflection

- If the object is a mirror, the apparent thickness of the glass that separates the reflecting surface and the surface of the mirror. A small gap between the reflection and the object will achieve this look.

- The angle of the reflecting surface to the location of the camera. The more oblique the angle the clearer the reflecting image.

-- Glows, Glints and Highlights

Most display programs have a built-in highlight generator, which is based on mathematical relationships between the location and orientation of the objects, the position of the light sources and the position of the camera. A highlight is basically a reflection of the light source. An extremely shiny surface will result in a highlight that is compact and will look like the light source. Most display programs assume that the light source is a point source; the source of the light is an infinitely small point located somewhere in space. A dull surface will reflect a larger yet more diffuse highlight. The problem herein lies that the light sources control the shading that is applied to the individual objects. It may be the case that a particular type of shading is desired for an object that conflicts with the location of light source necessary to provide the desired highlight. Or perhaps a glint may be needed to "spice up" the image. If the display program is capable of it, a second light source can be used to

create the desired highlight. This a fairly expensive solution to the problem because it requires the display program to calculate the shininess information for two light sources which involves a large amount of calculation time.

Another solution is to make a special object which "looks" like a highlight. This object can then be animated within the scene to the exact specifications of the animation. A highlight object can be made in a variety of shapes depending on the demands of the animation. The basic method that can be used for a highlight object is to create an object of the desired shape. A point should be defined to exist in the centroid of the object. A series of polygons should be created to connect the edges to the centroid point. (figure 9) Color assignments should be made to the individual points of the object. These color assignments are called "vertex colors" [22]. The transparency for each point can be individually assigned along with the color information. The points should be assigned such that the centroid point is opaque or nearly opaque while the edge points are transparent, or nearly transparent. The result of calculating the object will be a fuzzy

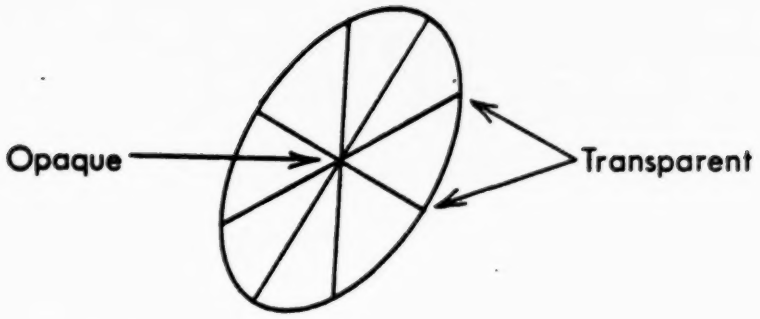


Figure 9.

Highlight object.

shape that is opaque near the center but gradates out towards the edges. This object can be animated to cover the surfaces of objects and displayed over those surfaces which are the reflecting surfaces, just as in the reflection model described above.

Glows for objects can be made in much the same way. The glow that surrounds the moon in the image described in Chapter 3 (figure 2) was attained using this method. Another method for creating glows is to blur an object

in screen space using a summed-area table technique developed by Frank Crow [6]. Once blurred, the image should be shown in the desired color of the glow. The object which is glowing can then be shown on top of the blurred image. This technique saves calculation time since the glowing object and its glow are calculated simultaneously. Applying the blur as a post process is much less expensive than calculating it initially. Under most circumstances this technique should work satisfactorily. It may be necessary to create a glow object in the same way that a highlight object is created. This glow object can be blurred for additional softening.

Glows are quite effective for producing certain lighting effects. If an object is rendered to represent a light source, a glow may be added to enhance the feeling. For example, a slight glow around a neon sign would help make it look like neon. Oddly enough this method is effective because of an aberration of the medium of film and video. A filmed light source tends to have a slight halo. Audiences have come to expect filmic aberrations in images, and as such these aberrations tend to make the image look more "real".

-- Shadows

Shadows can be calculated algorithmically by the display program, however calculating shadows in this way is expensive. The effect of shadows can be created by using a variety of techniques depending on the situation involved. Simple drop shadows of objects can be created simply by showing the object offset from the original object in a transparent grey. This offset image can even be blurred slightly to give the impression of a soft shadow (a technique which is quite expensive when using algorithmic methods).

Another technique is to create special "shadow objects" which can be animated to approximate the way in which real shadows would be cast. The problem with using this technique is that it requires extra time on the part of the animator to get a correct looking approximation. It does however work quite well under controlled circumstances. In the case of "Trash" [8] a special purpose program was generated to produce realistic shadows (figure 10). Input parameters to the

original data generation program were used as input parameters to a second program which generated simple polygonal shadow objects. These were then shown over the background with transparency. The results were quite satisfactory. Since the shadow objects were very simple objects (less than 30 polygons) they rendered extremely quickly.

Figure 10.

Scene from "Trash".

CHAPTER 6

Natural Phenomena

Natural phenomena are often difficult to try to simulate using computer animation. Despite this it seems to be a popular goal to achieve. Many computer scientists and animators have successfully created images which represent naturally occurring phenomenon. These images have included representations of water, clouds, smoke, and fire.

To animate it effectively, it is important to have an understanding of the natural phenomenon in question, at least to the point of knowing how it looks. It is imperative that an animator approach the problem by first identifying those things which make the effect look a certain way. One must identify, the predominant characteristics of the desired effect.

In the case of fire, such as a campfire, it is important to look at what it looks like and what are its characteristics. One of its foremost characteristics

is that it is amorphous; its shape changes in a random fashion. This is a fairly common characteristic among naturally occurring phenomenon. The random quality of the flame seems to follow certain rules. The motion of the flame as it travels is random but it flows in an upward direction. It also seems as though flame is broken up into distinct shapes which change shape, color and transmittance as they travel upwards. Fire is partially transparent, it is possible to see through it. Fire also doesn't have distinct edges, they are soft and blurred.

Given these considerations it is possible to arrive at several conclusions about how to approach the rendering of fire. It will probably be necessary to utilize random functions in determining properties of the data. These properties include: color, shape, transparency and blurriness. It is also important to be able to control these properties as a function of time. Since fire travels in an upward direction, it is important that ways are given to have control of the basic shape, and the motion of the flames.

In order to attain these controls it became necessary to generate a program. A premise for the

program is that the final image would be blurred. This blurring would occur in the two dimensional screen space [6]. In this way a constraint is placed on the flame animation, in that it must be calculated as a separate layer and combined with the other elements of the animation. This constraint is a fairly minor one, however, as it is not very difficult to combine multiple layers into a single composite scene as has been shown in earlier examples. A feature of the program is that it is designed to allow for looping motion of the flame. This allows for the calculation of a relatively small number of data sets for a much longer sequence. It is important that the looping sequence (or cycle) must not be easily detected. Therefore one must carefully balance the computational expense with the data economy. This may not be a problem at all, particularly in cases where other things are happening within the scene.

The basic concept used in this approach is the following: there are a random number of pre-defined shapes moving upwards along a random number of paths which are spaced randomly within a given area. The color and transparency are also chosen randomly within a range as defined by the user. This object is then calculated

in three dimensional space. The two dimensional image is then blurred to be combined with the other elements in the scene.

In addition the program allows the animator to change parameters which control: the area of the flame (by specifying the flame center and a radius), the flame height, the number of random shapes, the number of frames in the cycle, the rotation of the shapes along the paths, the rotation of the paths, as well as the dispersion of the paths from a flame center. The ability to control all of these parameters makes it possible to animate a variety of effects. Some of the possible effects that this program is able to produce are fire, steam, smoke and leaves. It is also possible to produce other types of events, such as confetti, or a water fountain.

A simple model can produce undulating water, again through the use of a random function. The effect calls for an undulating surface of water, as one might find in a fairly large body of water on a calm day. Again it is important to think about the object that is being represented, and to use it as a way of getting clues about how it would be animated. If an object is floating

on the surface of the water, it appears to bob up and down, as well as back and forth as a crest of the wave passes over it. It seems as though an object rarely (unless the object is gliding on the surface) if ever changes its general position in water, or at least it does so very slowly. The waves in water are propagated as a function of the water changing shape. It is as if each particle of water bumps into the next particle of water and so on [9, 18]. In fact this is what happens in water. If the points which define the surface of the water were to move in this fashion, the surface itself would seem to move according to this principle.

This is exactly what the program does. First the surface of the water is defined, as if it were completely calm. Then each point on the surface of the water is moved in a small circle. The starting point around the circle is selected in a random fashion. The points all rotate in the same direction however. The result is a smoothly changing surface structure, which looks convincingly like water. It is as if each point that defines the surface is like an object which floats on the surface.

The effect is furthered by using a texture mapping

technique to apply an undulating reflection across the surface of the water. This again is achieved through intuitive means. To do this it is necessary to pre-calculate the reflection image (as was described earlier in this paper), and then instead of simply showing the reflection over the surface of the water it is texture mapped onto the surface. In looking at the reflection of an object on water, it becomes apparent that the part of the reflection that is closest to the eye is distorted more than the part of the reflection that is farther away. This then requires that the texture mapping technique that is used must have some information about the relative position of the object in the screen, relative to the eyepoint. As a result the texture mapping is done as a pre-process to the display program but must have much of the information that the display program receives. The information that is needed is the distance that the point is from the eye, the surface normal of the point, and the screen space location of the point. Once this information is known the texture can be manipulated quite freely, and with a lot of knowledge about how it will appear on the screen [19].

The way in which the reflection was achieved in this scene was to take the screen position of each point and to assign this as the texture coordinate for an un-perturbed surface. The result would be the reflection for a perfectly calm surface. It would look like the surface was a mirror. However the texture coordinate is changed as a function of the distance of the eye as well as the orientation of the surface normal. The X and Z components of the surface normal are applied as an offset to the texture coordinate in the X and Y coordinates respectively of the screen space. This application is an inverse function of the distance that the point is from the eye. In short, this means that as the distance increases the offset is lessened. This value is then multiplied by a user defined value, which amplifies, or reduces the overall effect. The result is added to the the original texture coordinates. These texture coordinates and the object are then passed to the standard display program which renders them. The result is an undulating surface which is a function of the shape of the water surface.

While this technique really has very little basis in reality as to the way in which a real reflection

would be reflected from the surface (as it would be using a ray-tracing technique) the results are quite satisfactory. The technique also is quite computationally inexpensive since it does not use any expensive rendering methods, but uses information which is calculated quite simply.

-- Random functions and methods of control

Random functions are quite useful when a large amount of detail is needed for a scene. Random functions are a necessary part of being able to produce satisfactory natural phenomenon images [7]. However these random variables need to be controlled in some way. Animators should be able to have control over the way in which a random function is influencing their animation. The simplest way of controlling the way in which a random function occurs is to limit its range in some way. This can be done by using a multiplier for the function that the random number returns. In other words if the random number generator returns a value in the range of 0 and 1, then a multiplier can be used to bring the random number to within the desired range.

This range can be applied as a deviation from a mean or it can be used as an absolute value. A measure of control that animators can be given is the amplitude of the multiplier. They have control over the range of the random number generator. If the random number is applied as a deviation from a mean, then giving the multiplier a value of zero, would result in no random variation.

A more sophisticated way of controlling random numbers is to allow the user to define a probability curve. The way in which this can be instituted is to define the curve as a graph of value vs. probability. The area under the curve represents the events which are effective. Those points above the curve are ignored [20]. Take an example of a curve which controls the location of grass blades within a given area. Suppose the example calls for a bare spot to occur in the area of grass. The curve then can control the probability that a blade of grass will exist if it is within a certain distance from the center of the bare spot. Given a random distribution of blades of grass, the curve then determines whether that blade of grass is added to the data set or ignored (figure 11). A counter can be used to count the number of blades that are added to the

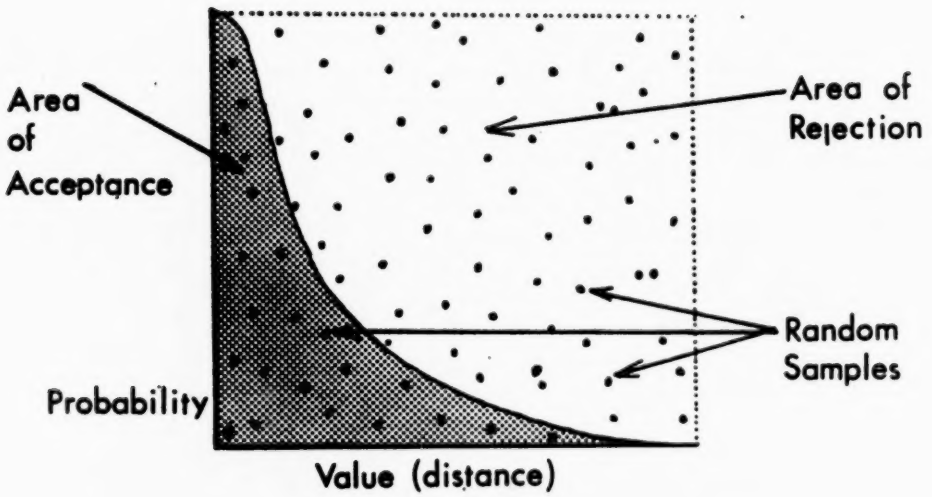


Figure 11.

Probability curve and random distribution.

data. The results of using this method is that the percentage total of the number of grass blades are distributed according to the curve which is user defined. While the distribution itself is random, a strict control is available to the animator. In the language of statistics this curve is the population distribution which is defined to be: "the frequency with which all units or observations in a population would be assigned or expected in each category or point on a measurement scale." [32]

CHAPTER 8

Conclusion

"It's possible to do darn near anything if we figure out certain definite things... We can do anything we think of with this."

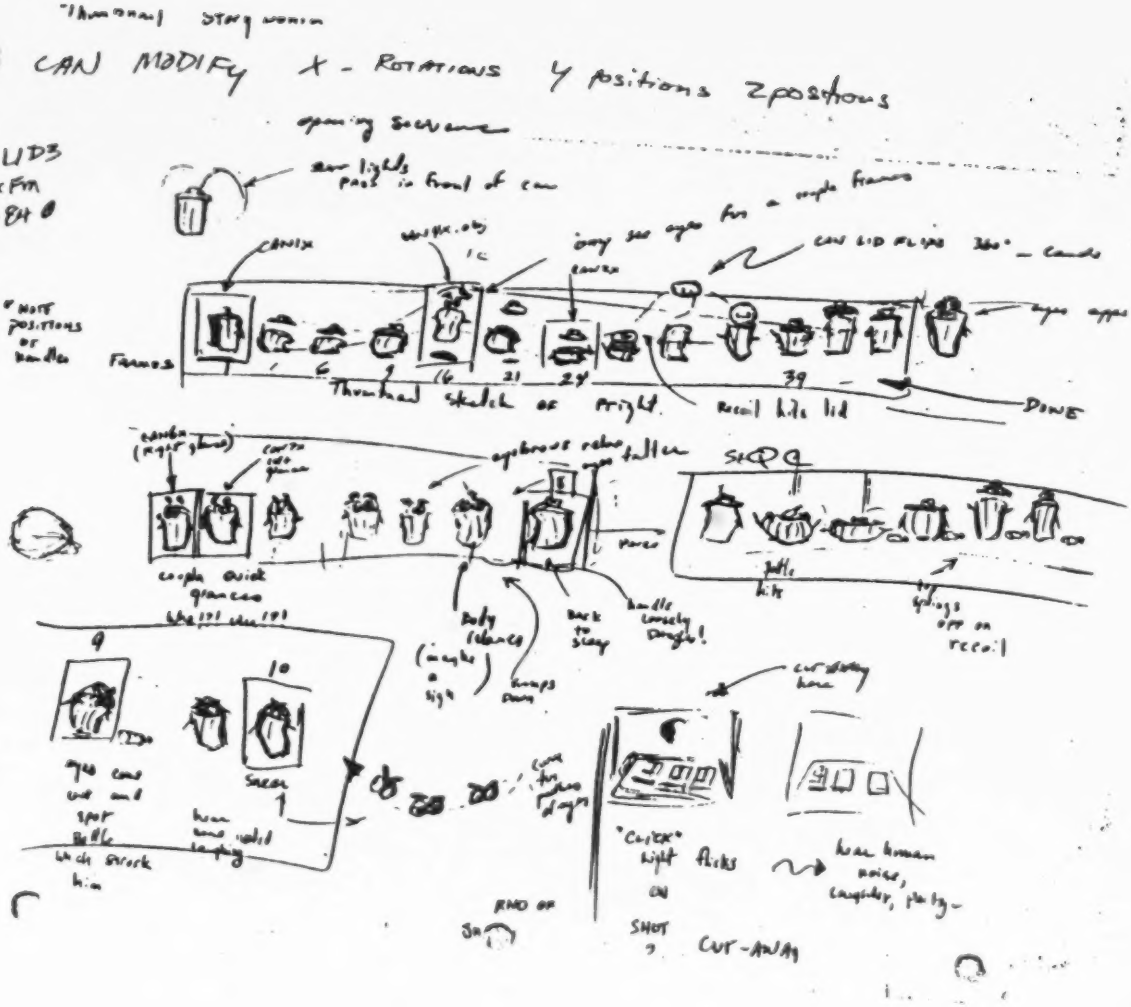
Walt Disney

from The Illusion of Life [24].

The quotation above says it all as far as the job of the animator is concerned. Computer animators must have the ability to solve a never-ending series of problems. They must have the ability to recognize those problems that need to be solved. The goal in solving the problems is to continue on; to develop the animation to the potential that is within it. In the development of animation tools and techniques it is the animation that drives the animator forward. The animator needs to be able to find solutions which are within the limits of the resources that are available, for if the solutions are unattainable they are not really solutions. There

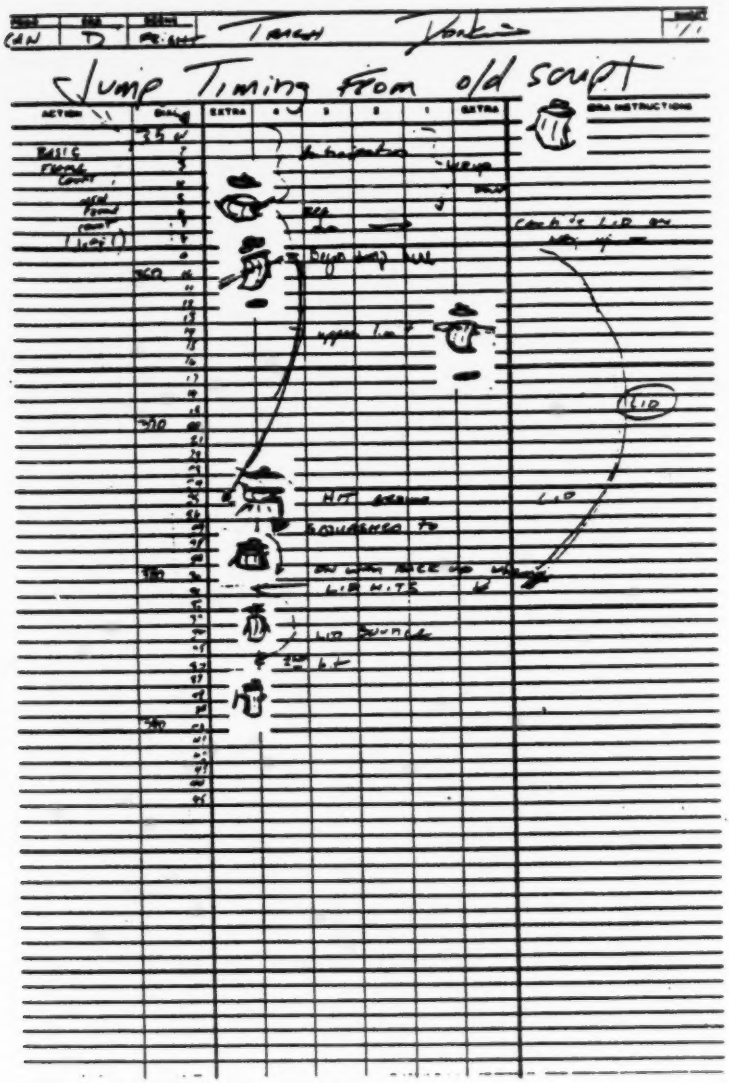
are always system constraints and there always will be. As such there will always have to be alternate solutions found to complex problems. These solutions need not result in lowered quality in the final product, but can instead be used to forge new and innovative techniques.

The goal of an animator should be to push to the limit the resources which are available to them, in an effort to produce the best possible animation. If these limits are too restrictive the animator should find ways to expand the limits. The most valuable tool for achieving this is to carefully analyze all of the aspects of the problem to formulate a solution that is as clean, simple and efficient as possible.


RB CAN MODIFY X - ROTATIONS Y positions Z positions
opening sequence
LIDS
CANIX
Thumbnail sketch of Bright
Recall kids lid
DONE
Baby reaches
back to sleep
"Cuter" light flicks
CUT-AWAY

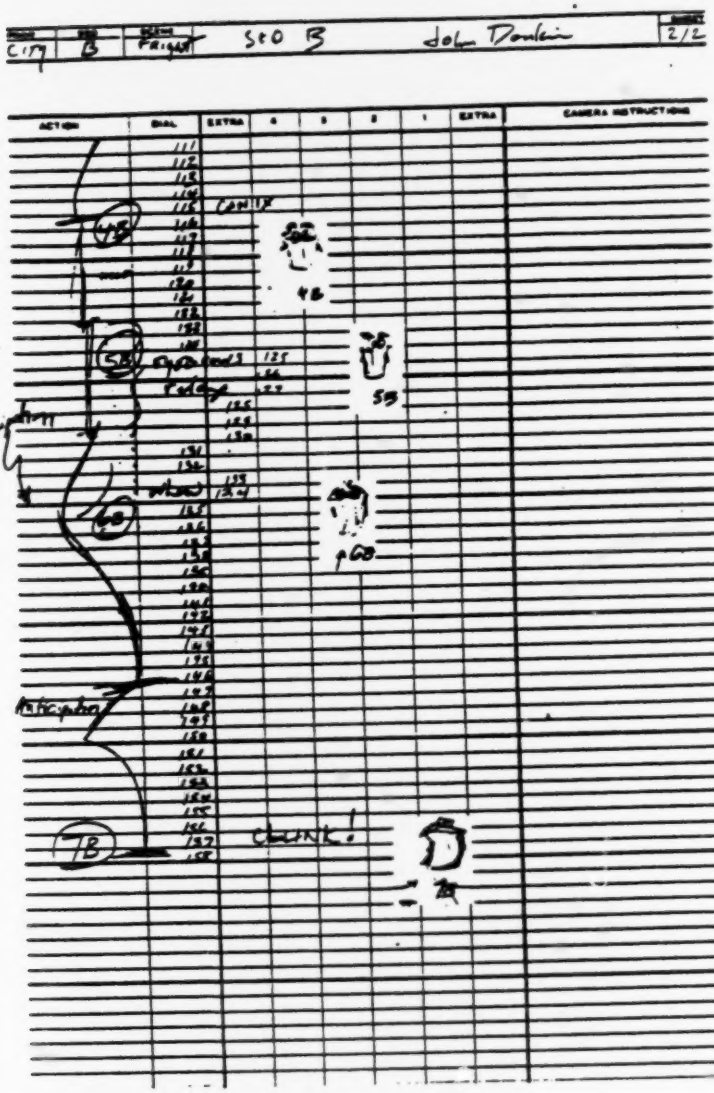
APPENDIX B

Timing Sheet



APPENDIX C

Timing Sheet



LIST OF REFERENCES

[1] Bergeron, Phillipe et al., "Tony DePeltrie", an animated film, Produced at The National Film Board of Canada, 1985 Siggraph Video Review, 1985.

[2] Blinn, James. "The Chinese Art of Chi-Ting", ACM Siggraph Tutorial Notes "Image rendering Tricks, tutorial number 12", Jet Propulsion Laboratory, California Institute of Technology. July 23, 1985.

[3] Blinn, James, "Computer Display of Curved Surfaces", An Unpublished Doctoral Dissertation, University of Utah, Dec. 1978.

[4] Blinn, James. "Systems Aspects of Computer Image Synthesis and Computer Animation". ACM Siggraph Tutorial Notes "Image Rendering Tricks, tutorial number 12", Jet Propulsion Laboratory, California Institute of Technology. July 23, 1985.

[5] Chuang, Richard, "RENDERING FOR TELEVISION", ACM Siggraph Tutorial Notes "Image Rendering Tricks, Tutorial Number 12", Pacific Data Images Inc. July 23, 1985.

[6] Crow, Franklin C., "Summed Area Tables for Texture Mapping", Computer Graphics, 18, (3), (July 1984), ACM Siggraph '84 Proceedings.

[7] Csuri, Charles A. et al., "Panel: The Simulation of Natural Phenomena", Computer Graphics, 17, (3), (July 1983), ACM Siggraph '83 Proceedings.

[8] Donkin, John C., "Trash", an animated film, produced at The Ohio State University, 1984 Siggraph Video Review, 1984.

[9] Fournier, Alain and Reeves, William T., "A Simple Model of Ocean Waves", Computer Graphics, 20, (4), (August 1986), ACM Siggraph '86 Proceedings.

[10] Gomez. Julian E., "Twixt: A 3D Animation System", 1984 Eurographics Proceedings. Copenhagen September, 1984.

[11] Greene, Ned, "Environmental Mapping and Other World Projections", IEEE CG&A, (pp. 21 - 29), November 1986.

[12] Ho, Hsuen-Chung, sphere-scan, a sphere rendering display algorithm. Developed at Cranston/Csuri Productions.

[13] Kingsbury, Douglas Lee, "Computer Generated, Three Dimensional Character Animation: A Report and Analysis", Unpublished M.A. Thesis, The Ohio State University, 1985.

[14] Kochanek, Doris H. U. and Bartels, Richard H. "Interpolating Splines with Local Tension, Continuity, and Bias Control". Computer Graphics, 18, (3), (July 1984), ACM Siggraph '84 Proceedings.

[15] Longtin, Tom, personal discussion.

[16] Girard, Michael and Maciejewski, A. A. "Computational Modeling for the Computer Animation of Legged Figures", Computer Graphics, 18, (3), (July 1984), ACM Siggraph '84 Proceedings.

[17] MacDougal Paul D., "Generation and Management of Object Description Hierarchies for the Simplification of Image Generation", A Report to the National Science Foundation, Division of Mathematical and Computer Sciences. The Computer Graphics Research Group, The Ohio State University, August, 1984.

[18] Peachey, Darwin R., "Modeling Waves and Surf", Computer Graphics, 20, (4), (August 1986), ACM Siggraph '86 Proceedings.

[19] Perlin, Ken. "An Image Synthesizer", Computer Graphics, 19, (3), (July 1985). ACM Siggraph '85 Proceedings.

[20] Reeves, William T., "Particle Systems--A Technique for Modeling a Class of Fuzzy Objects" Computer Graphics, 17, (3), (July 1983), ACM Siggraph '83 proceedings.

[21] Reynolds, Craig, "Computer Animation with Scripts and Actors", Computer Graphics, 16, (3), (July 1982), ACM Siggraph '82 proceedings.

[22] Sachter, Judy E., "The Basic Concepts of Three-Dimensional Computer Graphics for Artists", Unpublished M.A. Thesis, The Ohio State University, 1984.

[23] Schweppe, Marla Karen, "Timing and Movement in Three Dimensional Computer Animation", Unpublished M.A. Thesis, The Ohio State University, 1984.

[24] Thomas, Frank and Johnston, Ollie, Disney Animation: The Illusion of Life. Abbeville Press New York, 1981.

[25] Van Baerle, Susan, "Computer Generated, Three-Dimensional Character Animation", Unpublished M.A. Thesis, The Ohio State University. Dec 1985.

[26] Van Baerle, Susan, and Kingsbury, Douglas, "Snoot and Muttly", an animated film, produced at The Ohio State University, 1984 Siggraph Video Review, 1984.

[27] Van Baerle, Susan, "Uneven Bars", an animated film, produced at The Ohio State University, 1983 Siggraph Video Review.

[28] "Wavefront", Commercially Available Software, Wavefront Technologies, Los Angeles, California.

[29] Whitted, Turner, "The Hacker's Guide to Making Pretty Pictures", ACM Siggraph Tutorial Notes, "Image Rendering Tricks, tutorial number 12", Department of Computer Science, The University of North Carolina at Chapel Hill. July 23, 1985.

[30] Williams, Lance. "Pyramidal Parametrics", Computer Graphics, 17, (3), (July 1983), ACM Siggraph '83 Proceedings.

[31] Carlson, Wayne E., "Techniques for the Generation of Three Dimensional Data for Use in Complex Image Synthesis", A Report to the National Science Foundation Division of Mathematical and Computer Sciences. The Computer Graphics Research Group, The Ohio State University, Sept. 1982.

[32] Williams, Frederick, Reasoning With Statistics, Second Edition, Holt, Rinehart and Winston, New York, 1979.